Using Complex Event Processing for Dynamic Business Process Adaptation

Gabriel Hermosillo, Lionel Seinturier, Laurence Duchien

Using Complex Event Processing for Dynamic Business Process Adaptation

Gabriel Hermosillo, Lionel Seinturier, Laurence Duchien
INRIA Lille Nord Europe - University of Lille 1
Laboratoire LIFL - CNRS UMR 8022
Lille, France
Email: firstname.lastname@inria.fr

***Abstract*—As the amount of data generated by today's pervasive environments increases exponentially, there is a stronger need to decipher the important information that is hidden among it. By using complex event processing, we can obtain the information that really matters to our organization and use it to improve our processes. However, even when this information is retrieved, business processes remain static and cannot be changed dynamically to adapt to the actual scenario, diminishing the advantages that can be achieved. In this paper we present CEVICHE, a framework that combines the strengths of complex event processing and dynamic business process adaptation, which allows to respond to the needs of today's rapidly changing environments. We use a simple car rental scenario to show how CEVICHE could be used to maintain the quality of service of a business process by adapting it according to the situation.**

***Keywords*-Complex Event Processing; BPEL; process adaptation; QoS**

I. Introduction

Given the dynamicity of today's business environments, there is a need to continuously adapt the business processes in order to respond to the changes in those environments and keep a competitive level. One of the main concerns for on-line applications is to keep a high *Quality of Service* (QoS), for which they need to keep a constant monitoring of their processes. By using *Complex Event Processing* (CEP) we can facilitate the solution of this problem by gathering information about the different steps of the processes in order to determine whether a situation of low QoS is approaching. CEP is an emerging technology which allows to find real-time relationships between different events using elements such as timing, causality, and membership in a stream of data in order to extract relevant information [1].

CEP can be used, for example, to prevent the theft of merchandise from stores by creating relationships between the amount, kind, and movement of the products inside the store and sending an alert when a suspicious situation is detected [2]. However, there are some occasions in which it is not enough just to be able to obtain this information from simple raw data. For example, when monitoring the QoS, we could alert the administrator when the process is not responding as expected, but an optimal response would be to automatically adapt the business process according to the new context in order to continue in an optimal way, and this is why we developed CEVICHE (*Complex EVent processIng for Context-adaptive processes in pervasive and Heterogeneous Environments*).

The purpose of CEVICHE is to create context-aware business processes that are able to adapt dynamically in order to respond to different scenarios. CEVICHE relies on BPEL, since it is the most common orchestration language, it is an OASIS standard and it is an execution language and not a modeling language (like BPMN), and in CEVICHE the adaptation of the business process happens at runtime during the execution [3]. The decisions of how to respond to a specific scenario are done by collecting data from different sources and transforming it into useful information, using CEP. By using an aspect-oriented approach, we can define alternative processes that can be woven into the business process at runtime, allowing the business process to adapt in a dynamic way.

In this paper we use an on-line car rental application to show how CEP, and specially CEVICHE, can be used to maintain a high QoS by monitoring the business process and adapting it accordingly to respond to the context information gathered by the system.

The objectives in this paper are:

- To show why dynamic adaptation is needed in today's business processes.
- To integrate CEP into business processes to help in the decision making task.
- To provide a framework that facilitates such integration by giving the users a unique entry point to create dynamically adaptable business processes.

The rest of this paper is organized as follows. In Section II, we use a scenario to illustrate the motivation and challenges of our proposal. Section III presents a background of the different domains used in this paper. Section IV explains the CEVICHE framework and its architecture. In Section V, we discuss our proposal and present some validations. Section VI presents some of the related work. Finally, section VII concludes and discusses some future work.

II. Motivation and Challenges

In this section we present our motivation by using a small car rental scenario in which we want to monitor the QoS.

After that we present the challenges that we face when using static business processes.

A. Motivation

To show how CEP and CEVICHE can be used to maintain a high QoS we present an on-line car rental service. In this service, the client goes through a process of eleven steps to get a car, as shown in Fig. 1. The client starts the process by providing a valid license number, then selects the characteristics of the car to rent and finally pays and receives a confirmation from the system.

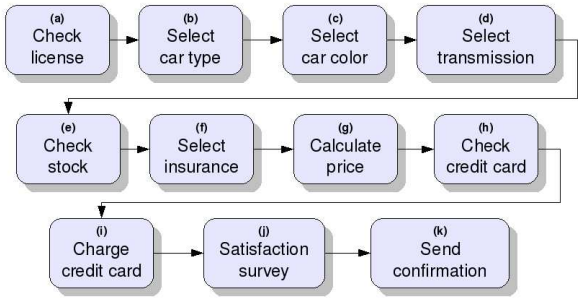

Figure 1. The car rental process

Whenever the application is getting a considerable amount of traffic, that could decrease the response time of the servers, the owners of the on-line application want to avoid the invocation of some optional tasks of the business process. This will help to maintain the QoS and allow the user to complete the car rental process in fewer steps and to spend less time waiting for the application to respond.

The optional tasks that could be excluded from the business process of this scenario, without altering the main objective, are: the color select (step c) and the satisfaction survey (step j). To skip them, an additional path from the preceding activity to the next will have to be set. Then, the process will have to be redeployed and the application restarted, for the changes to be considered.

In this scenario, the QoS is considered with two parameters: service performance and service availability. The performance of a web service can be measured by considering the time it takes to respond to a user query, while the service availability can be simply measured by the existence or not of a response from the service.

B. Challenges

When we want to monitor the business process and adapt it accordingly to respond to current situation, we face several challenges that make it difficult to accomplish.

Challenge 1: The first challenge we have in this scenario is the lack of specification in the BPEL standard to monitor the business processes, thus leaving each BPEL engine implementation to decide whether or not to include a monitoring interface [3]. But even when the BPEL engine used provides a monitoring interface, it does not necessarily mean that it will provide the precise information that we want to monitor.

Challenge 2: As our next challenge, we face that even when we could monitor the business process, the identification of a situation that needs the process to be adapted (like a low QoS), will add a lot of unnecessary code to the core business process definition, and in some cases will be impossible to specify using only BPEL.

Challenge 3: The last challenge that we face when using BPEL engines is that the process definitions are static, which means that we cannot adapt the business process without redeploying it, thus generating a downtime of the system and losing all the information of the current transactions. The only changes possible at runtime are the bindings to partner links, but they have to be previously defined at deploy-time and we cannot add new partner links at runtime [4].

III. Background

In this section, we present a brief introduction to the four main domains addressed in this paper: complex event processing, business process execution language, quality of service and aspect oriented programming.

A. Complex Event Processing

CEP is an emerging technology for finding relationships between series of simple and independent events from different sources, using previously defined rules [1]. The CEP technology can be used, among a lot of other things, to enrich the enterprise's existing processes, by introducing rules that will allow the capture of relevant information from the different steps of their business process [5].

For example, let us consider the scenario of a retail store that keeps a record of its inventory in an existing *Enterprise Resource Planning* (ERP) system and wants to keep a live monitoring of its stocks in order to prevent shortage. To achieve this, the store installs a CEP engine that will monitor the products movements through their life cycle in the store process by receiving and analyzing all the events generated by every change in the state. Since the objective is to monitor inventory, the CEP engine will only keep the events related to changes in the inventory and forget about the rest. By creating the necessary CEP rules, the configuration is set to specify the lowest acceptable stock of product that the store can have to avoid a shortage, *e.g.*, a 10% for normal products and a 5% for some low-demand products. Whenever a product reaches a minimum, the CEP engine alerts the managers so they can make a supply order.

In addition to that, CEP can also be used to predict unexpected situations. To complement the previous example, we can say that because of a global pandemic alert, hand sanitizers are very popular and are selling a lot more than usual. Given this demand, the store will run out of hand sanitizer before they can resupply it, even with the minimum

stock alert. By adding some specialized CEP rules to analyze the frequency of sells of each product during the last 4 or 5 hours, the engine could polarize these values to know in advance (if the sells rates are kept) that it will need to resupply before the expected time, which will allow them to react in time even before it reaches the minimum level.

B. Business Process Execution Language

The *Business Process Execution Language* (BPEL) is an XML-based language for composing services, created by IBM, BEA Systems and Microsoft in 2002, and later approved as an OASIS Standard as WS-BPEL 2.0 [3]. There are two types of service composition: orchestration (execution) and choreography (control). BPEL is an orchestration language, which means that it focuses on the flow of control and data among the different services of the business process, rather than on the specification of peer-to-peer collaboration.

BPEL uses web services as a way to communicate with the different parties involved in the business process. It has two types of activities: primitive and structured. The former refers to atomic or single activities while the latter refers to composite activities (a combination of several activities). Some instructions like `invoke`, `receive` or `assign` refer to the primitive activities, while `sequence` and `flow` are part of the structured activities and refer to the order in which the activities will execute.

In order to interact with the different parties of the business process (called `partners`), we need to define a partner link, which specifies the roles of the partner and the caller. We also need to define the different input and output variables that we will use to send information to and receive information from the service. Finally, BPEL also provides some facilities for transaction and exception handling.

C. QoS in Web Services

The goal of the Web Services effort is to achieve interoperability between applications by using Web standards. Web Services use a loosely coupled integration model to allow flexible integration of heterogeneous systems in a variety of domains including business-to-consumer, business-to-business and enterprise application integration [3]. Web services are the most popular technology to implement the service-oriented architecture and they use open Internet-based standards, like the *Simple Object Access Protocol* (SOAP) for data transmission, the *Web Services Description Language* (WSDL) for defining services, and BPEL for orchestrating services [6].

When these web services are used as part of a whole process, the QoS of the process depends on the QoS of the web services composing it. But, in order to use the services provided through the Internet by the different organizations, the users of those services need to know what to expect from them, in terms of QoS, so that they can offer a decent QoS to their own users.

The study of QoS for web services is not new, and there have been a good number works discussing how to estimate the QoS of a web service-based workflow [7], [8], [9], [10]. Also, in order to measure the QoS of the web services, many metrics have been proposed in the literature [7], [11], [12], [8], [13]. The QoS attributes can be classified as deterministic or non-deterministic. The former means that the attributes are already known before the execution of the service (*e.g.*, price, server location). The latter refers to the attributes that are unknown before the execution (*e.g.*, response time, availability). The monitoring of these attributes can be achieved in two ways: server-side monitoring or client-side monitoring. Since the user of the services does not always have the control of the hosting servers, we will focus on the client-side monitoring for our scenario.

As we mentioned in Section II, our scenario measures the QoS using two parameters: performance and availability. To calculate those values we will use the formulas provided by Oliver et al. in [11], as shown in Table I.

QoS Attribute	Formula
Performance	$\frac{1}{\#requests} \sum requestTime_i$
Availability	$1 - \frac{uptime}{downtime}$

Table I
QOS ATTRIBUTES

D. Aspect Oriented Programming

The domain of *Aspect-Oriented Programming* (AOP) appeared in 1996 [14], [15]. It was pioneered by Gregor Kiczales and his team, then at the Xerox Palo Alto Research Center. While original and innovative, the domain of AOP inherits results from other programming approaches, such as reflection, open implementations, meta-object protocols, and generative programming.

AOP, as a new programming paradigm, introduces notions such as aspect, join point, pointcut and advice code. However, these notions do not replace existing ones, such as class, object, procedure or method. Rather, AOP must be seen as a complement to these existing techniques. Furthermore, these notions are not specific to a programming style (*e.g.*, object-oriented or procedural) or a given syntax (Java, C#, Ada, COBOL, etc.). Aspect-oriented extensions exist for many languages, object-oriented or procedural, and in this case, BPEL.

AOP has been proposed as a technique for improving the separation of concerns in software systems and for adding crosscutting functionalities without changing the business logic of the software. AOP provides specific language mechanisms that make it possible to address concerns, such as

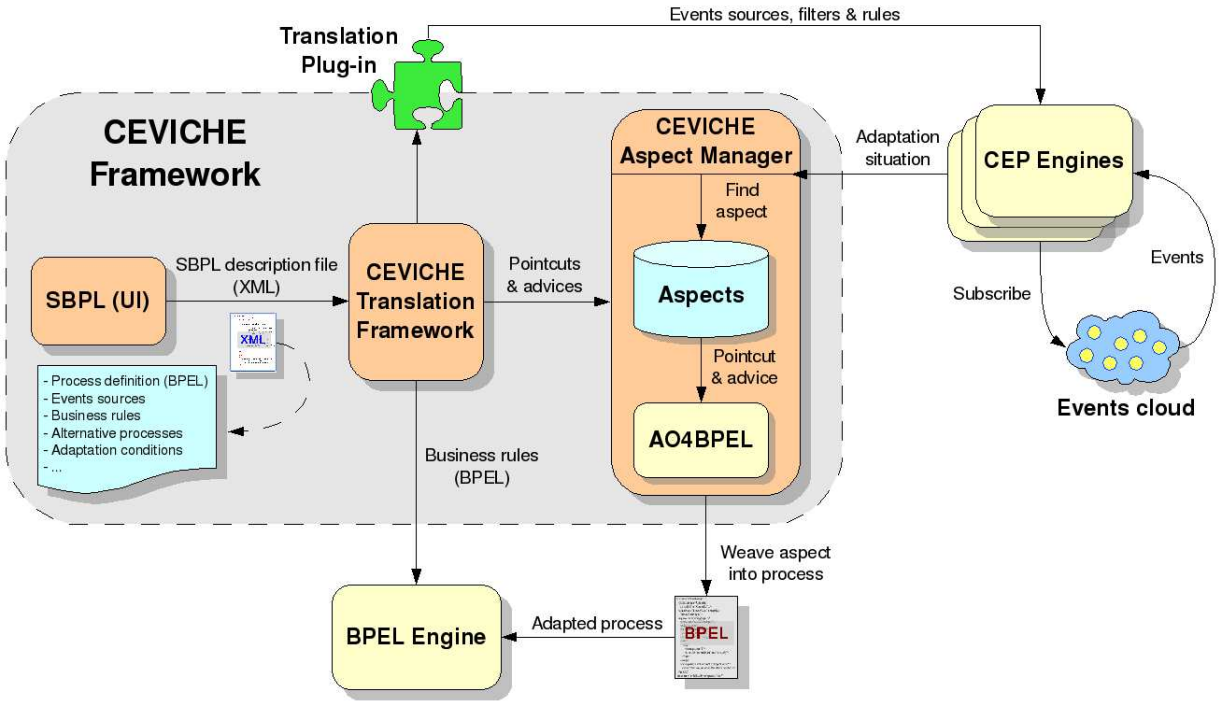

Figure 2. The CEVICHE framework

security, in a modular way. AOP languages and tools can be applied at compile-time or at run-time, this giving the designer the flexibility to use them in the most appropriate moment.

IV. The CEVICHE framework

In this section we present the CEVICHE framework. We start by giving an overview of the system, then we present the architecture and finally we explain how the process adaptation is realized.

A. Overview

CEVICHE is a framework that intends to facilitate the integration of CEP into existing business processes and to allow these processes to be dynamically adapted to different circumstances. With this framework we want to address mainly four issues: adaptation, dynamicity, integration to business process, and non-dependency to a specific CEP engine.

To address the first issue, adaptation, we use an approach based on AOP [14]. This approach allows the system to add or change services from the current business process and facilitates the task of separating concerns. To do this, AOP needs to know in which part of the business process it needs to perform the adaptation (the pointcut). Using the pointcuts, we can tell the AOP framework what special behavior (the advice code) we want to apply in that part of the business process. Once the advice code is woven, the process is adapted.

By analyzing the current events with CEP and using context information, CEVICHE can automatically decide when and how to adapt the system. This adaptation can be done at runtime, thanks to the advantages of using an aspect-oriented approach, thus giving a solution to the dynamicity issue.

To integrate BPEL with AOP, we use the AO4BPEL framework [16], which creates a wrapper around the BPEL interpreter and has the ability to weave the aspects at runtime to the business process. The advantage of using AO4BPEL is that we can change the business process specifications at runtime without the need to redeploy them, avoiding to lose all the ongoing transactions by doing that. In CEVICHE we integrate the AO4BPEL framework with CEP, allowing us to give a solution to the third issue.

Finally, CEVICHE aims to be able to work with any CEP engine available. For that, as part of this framework, we define a language called the *Standard Business Process Language* (SBPL), which gathers all the information about the processes, contextual environment, business rules, and adaptation conditions. This information is saved in an XML file that CEVICHE translates to the chosen CEP engine using the corresponding translation plug-in of that engine. This approach allows the users to define their business processes only once and deploy them using their preferred CEP engine.

```
<aspect name="MonitoringAspect">
  <partnerLinks>
    <partnerLink name="PerformanceMonitor" partnerLinkType="PerformanceLink"
                 myRole="caller" partnerRole="monitor"/>
  </partnerLinks>
  <variables>
    <variable name="activityStart" messageType="activityStartName"/>
    <variable name="activityStartTime" messageType="startTime"/>
    <variable name="activityStop" messageType="activityStopName"/>
    <variable name="activityStopTime" messageType="stopTime"/>
  </variables>
  <pointcutandadvice>
    <pointcut name="performance" contextCollection="true">
      //invoke[]
    </pointcut>
    <advice type="around">
      <sequence>
        <assign>
        <copy>
          <from variable="ThisJPActivity" part="name"/>
          <to variable="activityStart" part="activityName"/>
        </copy>
        </assign>
        <invoke partnerLink="PerformanceMonitor" portType="PerformanceLink"
                operation="startTimer" inputVariable="activityStart"
                outputVariable="activityStartTime"/>
        <proceed/>
        <assign>...</assign>
        <invoke partnerLink="PerformanceMonitor" portType="PerformanceLink"
                operation="stopTimer" inputVariable="activityStop"
                outputVariable="activityStopTime"/>
      </sequence>
    </advice>
  </pointcutandadvice>
</aspect>
```

Figure 3. The performance monitoring aspect

B. CEVICHE Architecture

CEVICHE is composed of three main parts: a user interface to create the SBPL files, a translation framework to manage the plug-ins for each CEP engine, and an aspect manager to deal with the process adaptation. CEVICHE also relies on different technologies to achieve the process adaptation, as shown in Fig. 2.

To configure the system, the user is provided with an interface to capture the different elements needed to adapt the processes, which are then saved to an SBPL file. The SBPL is an extension of BPEL which allows the user to include, in the business process definitions, the adaptation points and conditions in order to create dynamically adaptable business processes. The information in the SBPL file is sent to the translation framework, which separates the data in three parts: the business process (BPEL), the adaptation situations (CEP rules) and the aspects to adapt the process. Since there is no standard to define the CEP rules, the translation framework uses a specialized plug-in to send the adaptation information in the SBPL file in the specific CEP engine's format. This way, whenever the user wants to use another CEP engine, the only thing that needs to be done is to change the plug-in, without rewriting all the specifications of the business processes.

Once the initial setup is ready and all the components have been properly configured, the process starts and the information begins to flow from one component to the other, as seen in Fig. 2. First, the CEP engine subscribes to the different sources of events, here called the `events cloud`, which will provide the engine with the information it needs to take decisions and create complex events.

The CEP engine will gather all the events, filter the interesting ones according to the business rules and find relations that can generate complex events. When an adaptation situation is detected, the CEP engine notifies the aspect manager, which in turn searches for the corresponding aspect to adapt the business process. Once the aspect is selected, CEVICHE uses the AO4BPEL engine to weave the code into the process, adapting it at runtime.

C. Process adaptation

CEVICHE uses an aspect-oriented approach, which allows the system to add and remove functionality at runtime. When this approach is applied to the business processes, it allows them to be dynamically adapted.

With AO4BPEL we can handle two types of aspect deployments: process-level deployment and instance-level deployment [16]. The former is used when the aspect is needed in all the instances of the business process, while the latter is used when only a specific kind of instances are targeted.

The process can be adapted using three kind of advices: before advice, after advice and around advice. The first

one intercepts the call before the joinpoint, executes its task and then lets the process continue the normal flow, including the execution of the joinpoint activity. The after advice gets executed just after the joinpoint activity is completed and the process continues normally afterwards. Finally, the around advice intercepts the call just like the before advice, but it adds functionality before and after the join point activity. Moreover, this advice can be used without executing the joinpoint activity.

An example of an aspect definition can be seen in Fig. 3. In this example we see the use of the special reflexive variable ThisJPActivity (line 20), which refers to the join point (the intercepted activity). This variable can obtain all the information of the join point, in this case the name, so that it can be used by the advice.

We can also have a pool of different services that can replace the original one and we can even have a historical QoS rating for each one, that will allow us to select the best option. This pool of services can even include composite services, which can replace the original one.

V. Discussion and validation

In this section, we present how we can use CEVICHE to deal with the challenges presented in Section II. In a nutshell, we had three main challenges in our scenario: QoS monitoring, identification of special situations and dynamic adaptation.

Challenge 1: For the monitoring of the QoS, in our car rental scenario, we created an around advice. This advice is deployed at process-level, since we want to monitor all the instances of the business process. In the advice we include four additional activities, two before and two after the joinpoint. The first activity is used to count the number of times the joinpoint activity is called, then it calls the second activity which records the start time of the joinpoint activity. Once the joinpoint activity is executed, the third extra activity is called. This activity measures the duration of the invocation by subtracting the starting time from the current time, thus getting the performance of the activity for that invocation. The final activity is used to count the number of times the joinpoint has been successfully executed to calculate the availability. By using this two metrics (performance and availability) we can monitor the QoS of the business process.

Challenge 2: To identify the special situations where the process needs to be adapted, we created some CEP rules using the formulas presented in Section III-C. Whenever the performance of the car rental process dropped more than 25% or an activity was not available, the CEP engine notified CEVICHE.

Challenge 3: When CEVICHE received the notification about the drop in performance, it automatically deployed the instance-level aspect that skipped the two optional tasks of the car rental scenario: the color select (step c) and the satisfaction survey (step j). When the notification was about an unavailable service, then CEVICHE searched for alternative service from the pool. If an alternative service is found in the service pool, then an aspect deployed at instance-level to change the service. We tried this by turning off the insurance selection service (step f) in the car rental process and CEVICHE automatically changed to an alternative service of another provider. The adaptation of the business process was done dynamically and automatically, without loosing any information from the different instances of the process.

Performance: As it can be expected, the inclusion of aspects to dynamically adapt the process, as well as a CEP engine to discover the adaptation situations, require some additional time and resources. This overhead may vary depending on the number of rules that need to be processed by the CEP engine and the complexity of the aspects that will be woven. However, the overhead induced to the original process is negligible compared to the cost of the whole process, specially when dealing with Internet interactions with partners. The AO4BPEL engine adds only an overhead of 1% of the execution time, and could be even less when the processes are bigger, since the cost is mostly the same for the whole system and does not vary in terms of the number of activities. For the CEP engine, the overhead is also insignificant, taking it less than 1 ms to process each event through the whole set of rules. In this case we used the Esper engine, which is an open source stream event processing engine [17]. The efficiency of the CEP engine to process the events is such that it exceeds over 500,000 events per second.

VI. Related work

CEP and BPEL: As mentioned earlier in this paper, CEP is an emerging technology and the use of it in the business processes is a recent topic of interest and research [18], [19]. An analysis of scenarios of composite event patterns comparing BPEL and BPMN is done in [18]. The authors analyze patterns of events that go from conjunction and cardinality to time relations and event consumption possibilities. The conclusion of their study is that neither BPEL nor BPMN are capable of supporting complex event scenarios in their specifications, so there is a need to integrate event pattern descriptions into the process definitions language, but they do not mention the need to adapt the process according to those complex events.

In [19], we present a service to add traceability to the RFID tagged products by using Complex Event Processing. When the RFID events are captured, they are transformed into business events that correspond to the business rules defined in the process which allows the users to have a better understanding of the status of their products, *i.e.*, the product's location and the environment it has been exposed to.

Business process adaptation: The need to adapt a process has been a topic of interest in the recent years and there have been different approaches that offer solutions to it [20], [21], [22], [23]. In [20], the authors propose to deal with process adaptation by adding a web service repository that will handle the web services to invoke in each case. Whenever an invocation of a web service is done, the call is intercepted and the repository is checked for changes in the process definition, before the invocation of a web service. If there have been some changes, then it examines the available web services in the repository and chooses the one that best suits the criteria, otherwise the invocation is executed as usual.

The authors in [21] use an aspect-oriented approach introducing `executable models`, which are used to represent the cross-cutting concerns. They use `open objects`, which are representations of the state of the elements in the model, to monitor the invocation of services and adapt the process by weaving the interaction with other models before (activation) and after (deactivation) the call to the service.

Another aspect oriented implementation, using the Spring .NET framework, is presented in [22]. They use a contract-based approach to assign a web service to each instance of an execution call. To achieve adaptation of the process they can change the contract at runtime to assign a new web service for the call. They can also adapt an existing implementation of a web service by using aspects to weave the new behavior.

An adaptation of the BPEL language called VxBPEL is presented in [23]. The authors insist on the need of flexibility and variability in the service-based systems and the lack of them when deploying BPEL processes. They extend the BPEL language to add new elements like `Variation Points`, which are the places where the process can be adapted and `Variants`, which define the alternative steps of the process that can be used. VxBPEL also accepts new `Variants` to be added at runtime, allowing the systems to be adapted without redeploying the process.

BPEL context adaptation: The work that is closer to our proposal is the one presented in [24]. The authors present a plug-in based architecture for self-adaptive processes that uses AO4BPEL. Their proposal is to have different plug-ins with a well-defined objective. Each plug-in will have two types of aspects: the `monitoring aspects` that will check the system to observe when an adaptation is needed and the `adaptation aspects` that will handle the situations detected by the monitoring aspects. Whenever the conditions of a `monitoring aspect` are met, it uses AO4BPEL to weave the `adaptation aspects` into the process at runtime. In our approach we deal with the monitoring part using the rules deployed in the CEP engine, which will detect special situations (by relating simple events) and select the aspects to be used to adapt the process.

An advantage of their work is that the monitoring aspects can be hot-deployed to their BPEL engine while with our approach the changes in the rules might not be considered at runtime, depending on the CEP engine. On the other hand, this difference also shows an advantage for our proposal, since we are not tied to a single engine and we can use any CEP rules already defined for the monitoring process, while in their case we would need to create a new plug-in for each new situation we want to monitor. Also, even if we needed to restart the CEP engine in order to consider the new rules, this would not affect any active running processes, as it would if we restarted the BPEL engine.

VII. Conclusions and future work

Process adaptation and Complex Event Processing are two topics that are creating a lot of interest in the research community, however there is still no integration of both domains. In this paper we presented CEVICHE, a framework that intends to facilitate the integration of CEP into existing business processes and to allow these processes to be dynamically adapted to different circumstances. With CEVICHE we addressed four issues: adaptation, dynamicity, integration to business process, and non-dependency to a specific CEP engine. As part of the CEVICHE framework, we proposed the SBPL, an extension of BPEL that allows the user to include the adaptation points and conditions in order to create dynamically adaptable business processes. The SBPL uses special plug-ins to deal with the different languages of the CEP engines, allowing the users to write their process specifications only once and deploy it in the engine they want. Using a simple car rental scenario we showed how CEVICHE can be used to monitor the QoS of a business process and adapt it dynamically to keep a competitive level whenever the QoS dropped, without the need to redeploy the process and without loosing any current transactions.

Thanks to the modular architecture of CEVICHE, it is not strongly linked to any third party technology. Even though, for the moment we use AO4BPEL to deal with dynamic adaptation of business processes, we could change it for any other technology that allows us to do better dynamic adaptation, or even develop our own. This architecture also allows our work to be componentized in the future, facilitating the integration with other technologies and the interaction with the different parts of the architecture. We also plan to work on the definition of a RESTful architecture to leverage on the deployment of CEVICHE components and facilitate the evolution of the architecture by adding or changing components. We are still working with the SBPL in order to have a very vast coverage of the CEP needs, and for the moment we only have some basic support for SQL based CEP engines like Esper.

REFERENCES

[1] D. C. Luckham, *The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems*. Addison-Wesley Longman Publishing Co., Inc., 2001.

[2] N. Huber and K. Michael, "Minimizing Product Shrinkage across the Supply Chain using Radio Frequency Identification: a Case Study on a Major Australian Retailer," in *ICMB '07: Proceedings of the International Conference on the Management of Mobile Business*. IEEE Computer Society, 2007, p. 45.

[3] "OASIS Standard. Web Services Business Process Execution Language Version 2.0," http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html, April 2007.

[4] M. B. Juric, *Business Process Execution Language for Web Services BPEL and BPEL4WS 2nd Edition*. Packt Publishing, 2006.

[5] T. Ku, Y. Zhu, and K. Hu, "A Novel Complex Event Mining Network for Monitoring RFID-Enable Application," in *PACIIA '08: Proceedings of the 2008 IEEE Pacific-Asia Workshop on Computational Intelligence and Industrial Application*. IEEE Computer Society, 2008, pp. 925–929.

[6] M. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann, "Service-oriented computing: State of the art and research challenges," *Computer*, vol. 40, no. 11, pp. 38–45, November 2007.

[7] J. Cardoso, A. Sheth, J. Miller, J. Arnold, and K. Kochut, "Quality of service for workflows and web service processes," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 1, no. 3, pp. 281 – 308, 2004.

[8] M. C. Jaeger, G. Rojec-Goldmann, and G. Muhl, "Qos aggregation for web service composition using workflow patterns," in *EDOC '04: Proceedings of the Enterprise Distributed Object Computing Conference, Eighth IEEE International*. IEEE Computer Society, 2004, pp. 149–159.

[9] H.-C. Wang, C.-S. Lee, and T.-H. Ho, "Combining subjective and objective qos factors for personalized web service selection," *Expert Systems with Applications*, vol. 32, no. 2, pp. 571 – 584, 2007.

[10] L. Zeng, B. Benatallah, A. H.H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, "Qos-aware middleware for web services composition," *IEEE Trans. Softw. Eng.*, vol. 30, no. 5, pp. 311–327, 2004.

[11] O. Moser, F. Rosenberg, and S. Dustdar, "Non-intrusive monitoring and service adaptation for ws-bpel," in *WWW '08: Proceeding of the 17th international conference on World Wide Web*. ACM, 2008, pp. 815–824.

[12] M. Gillmann, G. Weikum, and W. Wonner, "Workflow management with service quality guarantees," in *SIGMOD '02: Proceedings of the 2002 ACM SIGMOD international conference on Management of data*. ACM, 2002, pp. 228–239.

[13] C. Patel, K. Supekar, and Y. Lee, "A qos oriented framework for adaptive management of web service based workflows," in *DEXA*, 2003, pp. 826–835.

[14] G. Kiczales, J. Lamping, A. Mendheka, C. Maeda, C. V. Lopes, J.-M. Loingtier, and J. Irwin, "Aspect-Oriented Programming," in *Proceedings of the European Conference on Object-Oriented Programming (ECOOP)*, ser. Lecture Notes in Computer Science, S. Gjessing and K. Nygaard, Eds., no. 1241. Springer, June 1997.

[15] G. Kiczales, E. Hilsdale, J. Hugunin, M. Kersten, J. Palm, and W. G. Griswold, "An overview of aspectj," in *ECOOP '01: Proceedings of the 15th European Conference on Object-Oriented Programming*. Springer-Verlag, 2001, pp. 327–353.

[16] A. Charfi and M. Mezini, "Ao4bpel: An aspect-oriented extension to bpel," *World Wide Web*, vol. 10, no. 3, pp. 309–344, 2007.

[17] EsperTech, "Esper," http://esper.codehaus.org/.

[18] A. P. Barros, G. Decker, and A. Großkopf, "Complex events in business processes," in *BIS*, 2007, pp. 29–40.

[19] G. Hermosillo, J. Ellart, L. Seinturier, and L. Duchien, "A Traceability Service to Facilitate RFID Adoption in the Retail Supply Chain," in *Proceedings of the 3rd International Workshop on RFID Technology - Concepts, Applications, Challenges IWRT 2009*. INSTICC Press, Portugal, 05 2009, pp. 49–58.

[20] F. A. A. Lins, J. C. dos Santos Júnior, and N. S. Rosa, "Adaptive web service composition," *SIGSOFT Softw. Eng. Notes*, vol. 32, no. 4, p. 6, 2007.

[21] M. Sánchez and J. Villalobos, "A flexible architecture to build workflows using aspect-oriented concepts," in *AOM '08: Proceedings of the 2008 AOSD workshop on Aspect-oriented modeling*. ACM, 2008, pp. 25–30.

[22] S. S. u. Rahman, N. Aoumeur, and G. Saake, "An adaptive eca-centric architecture for agile service-based business processes with compliant aspectual .net environment," in *iiWAS '08: Proceedings of the 10th International Conference on Information Integration and Web-based Applications & Services*. ACM, 2008, pp. 240–247.

[23] M. Koning, C.-a. Sun, M. Sinnema, and P. Avgeriou, "Vxbpel: Supporting variability for web services in bpel," *Inf. Softw. Technol.*, vol. 51, no. 2, pp. 258–269, 2009.

[24] A. Charfi, T. Dinkelaker, and M. Mezini, "A plug-in architecture for self-adaptive web service compositions," in *ICWS '09: Proceedings of the 2009 IEEE International Conference on Web Services*. IEEE Computer Society, 2009, pp. 35–42.